Articles

Adventures in Game Development / Pixelhumans

Introduction

The PixelHumans project is developing faster than I had planned, so I've established the Hashtag #Pixelhumans on Twitter. As long as this message is here, the article is work in progress (wip) and can change quickly at any time.

Who looked around on my web page, has looked certainly already briefly into my Cyberspace article. I had worked a little with the Unreal Engine 4 and implemented a small idea, which then ran aground. I wrote that it might be something until 2020, but the chances are rather bad. It was fun anyway. In the past weeks I started to create Pixel-Art and created a subpage for it. After that everything happened very fast and I had the idea to release my own little game demo. Since I had no idea in which direction this should go, I looked at the Unity software and a few tutorial videos on the Internet. Also this game engine was too big for me and I wanted to bake small breads. Through Twitter I found out about the Pico-8 Fantasy Console and bought it. With it you can work really well. I will add small updates here in this article from time to time, what I learn in the area of game development and what experiences I make. I don't always refer to Pico-8, but also to board games or game design as an art form. so this is *not* a tutorial and I assume that the content will be better understood by experienced people.

Pico-8

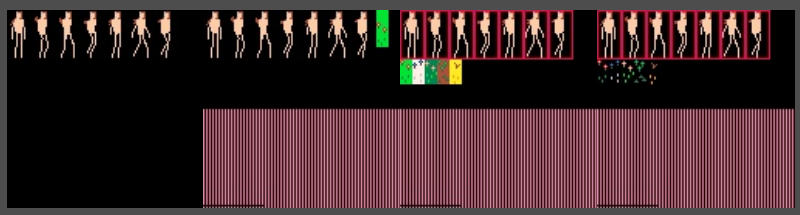
You can work very well with the Pico-8 Fantasy Console. You do everything alone for your game. The source code, the sprites, maps and the music. Everything is created and tested in the small virtual machine. The community (Search Twiter with #Pico8)is super nice and helpful if you ask questions sensibly. The API and FAQ are well documented and there are even some tutorial videos. What personally inspires me the most is the simple concept. You program in Lua, a lightweight, multi-paradigm programming language that is really easy to learn. For me, who was trained as a programmer, sometimes even too easy, because I already thought too complicated about a problem several times while programming. You can export your programmed game super fast for Linux, Mac, Window and HTML(web/Browser). It even automatically recognizes my XBox controller, without having to adjust anything. You can even play the HTML versions on your smartphone, because everything, really everything, is done automatically in the background for you. You don't have to worry about anything. You can publish your games as cartridge (a png picture). The complete game, music, maps and source code are in this little PNG and people with the Pico-8 console can download and play these cartridges.

If you don't like some colors of sprites you can simply change them, everything is open and in many cases you are free to edit them. But you can also keep your sourcecode and only publish the binarys or the HTMl e.g. on Itch or Gamejolt, but there is enough sourcecode you can look at to learn tricks or be inspired by ideas of other game developers. Some counterarguments are the maximum size for your game, which is 32K and that you only have up, down, left, right and two more buttons available. You also have to limit yourself to 16 colors. For me this is a challenge I like to work with, because you have to snap more. If you want to know what a good game can be, just have a look at Curse of the Lich King from Johan Peitz.

PixelHumans

Cartridges (V1-V4)

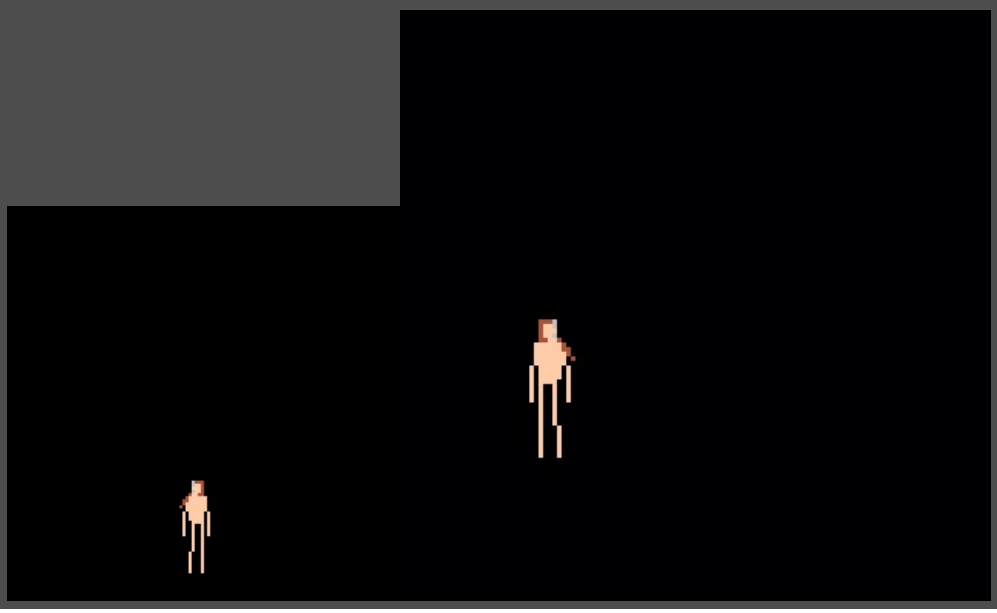
Spritesheets (V1-V4)



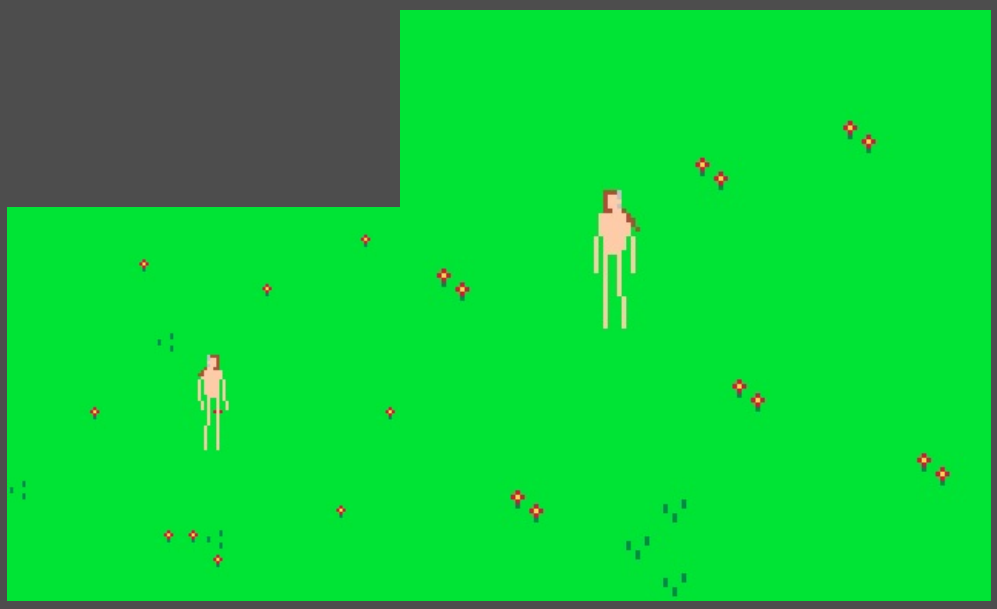
As i mentioned in the article One hour, one Live the idea for pixelhumans has been in the back of my head since 2014. of course this idea is not very original and if i had had more time for a private project in 2014, i could have taken the survival games trend with me. Now the market is overrun with Minecraft clones, Zombie or post war survival games. But that doesn't matter, because I don't stick to trends nor am I interested in what the *market* or the *players* find great and hip. Trends come and go and go when you consider how often and in how many variations zombies have appeared, I don't think much about the game concept. It's also something that the players already know and can integrate more quickly into the world or the game mechanics. Minecraft excelled with its very simple learning curve and appealed to a lot of players.

```
A human starts in the desert and has to find a stone as
fast as possible, to cut off a branch, run to a water
hole (drink), then run to a berry bush (eat). Then
find another stone with the stone, make a sharp stone
out of it, sharpen the stick with it and kill a rabbit.
```

After working through some tutorials, I decided to develop a very small demo for my first task. A small human should run over my screen. As always, you just imagine it and then you are surprised what unexpected problems arise. So I find the format of 8x8 pixels (tile) quite nice, but I really wanted it to be a little bigger. So before I programmed, I had to change the sprites again. I already had some ideas on my underside and adapted them for Pico-8. In theory I imagined it all quite well, but the result didn't convince me. If you look at the gif animation below, the legs are actually just a pulp of single pixels. This becomes exhausting after a few minutes, if you have to watch it all the time. So I have to reduce the number of tiles and see if I can adjust the speed of the tiles in the source code. All in all I'm very satisfied and it's worth half a day's work. It's also motivating if you can see the first results after a few minutes, because I've already made an experience with my work as a programmer. No matter if you develop medical software or computer games in your spare time, after a few days you automatically start to crunch and often have to kick your own ass to finish your project. But you shouldn't let that scare you off and I won't let you scare me, because working with the Pico-8 Fantasy Console is a lot of fun.



In the next step I have completely reworked the pixel human once again. I only had four tiles left, which didn't cause such a strong flickering. I also made the figure simpler. So that everything fits from the setting, I created a map with some flowers and blades of grass. Then the figure can run around and it looks better. I still have to deal with the colors, because the green is already very bright and the person does not stand out enough from the background.



Biomes

After I became friends with being able to work with only 16 colors, I thought about what kind of world the players can act in. Although the genre of survival games is already hard sucked, but for a small game this is still suitable. This brings us to the map and the biomes. I decided for grassland, tundra, jungle, swamp and desert. The red numbers are the numbers for the colors you can see in Pico8. This is similar to Minecraft and in some places I orientate myself. The main thing is to learn something. For all biomes I created small plants (flowers, grass or icy branches) so that the map doesn't look so boring. To create more variations and to make the play area more interesting you can flip single tiles. That goes in x or y direction. As I already mentioned above, we have very few resources available and we have to see how we can best invest them.

Who perhaps once in his life has played the computer game Minecraft or One hour one Life will notice the similarity. There were also [x] biomes. That was enough to bring a regular change into the game. Biomes were simply randomly generated by Minecraft and then the player could move around. If each of our biomes has two variations, we have a total of 10 basic tiles with which we can generate a map. The base of the map (the color) is created by a function that simply creates the entire map with one color. The individual tiles are then placed on this. However, the specific flowers etc. should already remain in the suitable biom. The player would have questions, for example, if an ice flower appeared in a desert. That's what many game developers forget. Even if you build an invented world, it should always have logical points in some places. You can interpret this very freely in many places, but you should not make it too difficult for the player to *settle in* to the game world.

Grass	Tundra	Jungle	Swamp	Desert
[64] Red flowers	[65] Ice flowers	[66] Pink flowers	[67] Green Plants	[68] Rotten bush
[80] Dark grass	[81] Ice things	[82] Light grass	[83] Swamp things	[84] Rotten grass

I have created a table here showing how I plan to do this with the biomes. I assign a unique number to all tiles. We find the number in the Pico8 development environment and help us with programming. It is very easy to program `MAP(64)` (as a green tile e.g.) into the source code and we can read it in the Pico8 API cheatsheet. Of course, this can get a lot more complicated, but I will only describe the simplest basic functions here. Since I want to randomly generate the biom later on, it will be quite different anyway. In addition, I have given the individual tiles a unique identifier. This helps me personally when the project gets bigger. You should be able to distinguish between `red flowers` and `pink flowers`. If you work some years as a programmer, you should always make a reference table in which you document related data. For example, if I later convert the identifier `red flowers` to `RFL`, I can read it here in the documentation if I should forget. For example, I already have `PLR` = player, `SPR` = sprite and `FLP` = flip in my code. There are strange rules in my head, so I always have to abbreviate all identifiers with three letters.

After a few days of doing a few experiments, I noticed certain things. If we look at the picture above (with the flowers and the background (above)), we can see that the individual grasses, flowers, sticks, etc., have the background of the biomes. That is ok, if we want to settle red flowers in the grass Biom. But if they were set in the jungle, we would have the background of the actual grass biome. We can avoid this if we set the background colours to black. In Pico8 black is the *blanc* color (if you didn't set it differently in your sourcecode). So it is not shown as black, but as the background color on which the tile lies. When I was younger, there were sometimes games that drew their sprites with a certain pink tone, if this should not be displayed in the game. I haven't found an exact foundation in the internet yet. I think the pink color was rarely used as a more regular color in a computer game. But this is only a guess. Nowadays you don't see that so often.

Since we want to save as much work as possible in our sourcecode we put the plants etc. into our `_init()` code. So we can access it quickly and don't have to search for it all the time. I have refined a rule already mentioned above. Each tile consists of two words which describe the content of the tile e.g. `red flowers`. Each identifier has three letters. Now we take from the first word, the first two letters and from the second word, only the first letter. This brings us to three letters. For red flowers this would be [re]d [f]lowers, i.e. `RFL`. According to this rule we can describe the remaining nine tiles. I have created a table to make this easier to understand. let's write our code as short as possible, but give people the chance to understand our syntax. NAT is for (NAT)URE.

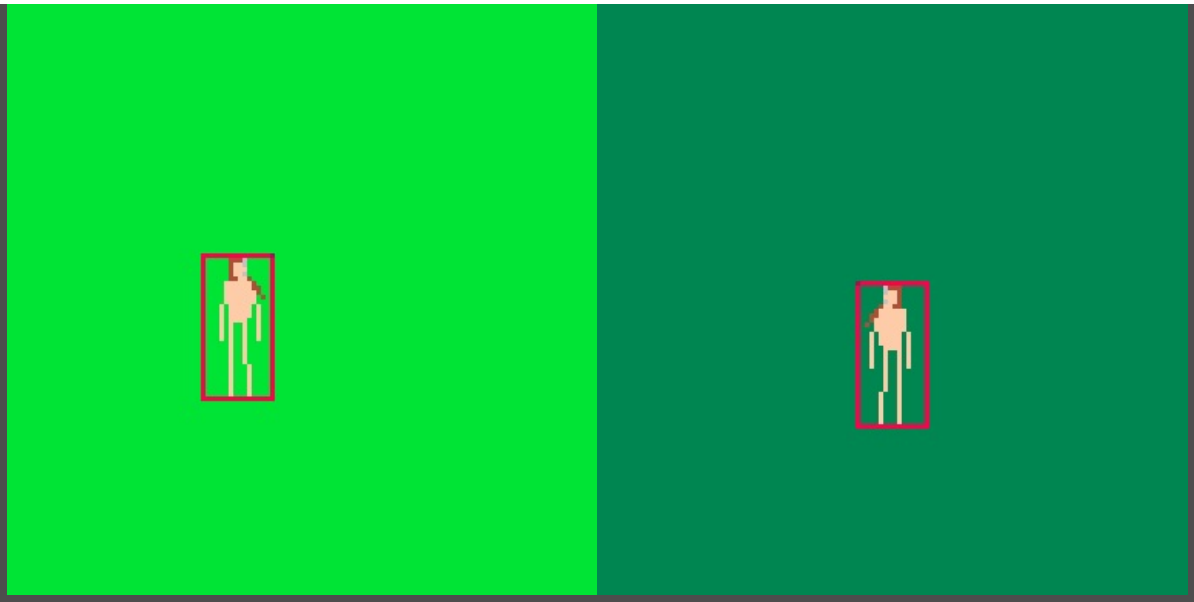
No.	Short	_init()	Description
64	REF	NAT.REF = 64	Red Flowers
65	ICF	NAT.ICF = 65	Ice Flowers
66	PIF	NAT.PIF = 66	Pink Flowers
67	GRP	NAT.GRP = 67	Green Plants
68	ROB	NAT.ROB = 68	Rotten Bush
80	DAG	NAT.DAG = 80	Dark Grass
81	ICT	NAT.ICT = 81	Ice Things
82	LIG	NAT.LIG = 82	Light Grass
83	SWT	NAT.SWT = 83	Swamp Things
84	ROG	NAT.ROG = 84	Rotten Grass

Enhancements

A challenge with Pico8 is that you have very little disk space available. This applies to the lines of code as well as to the tiles we use to display graphics. In the *Biome* chapter I have shown the base areas of the biomes as tile. I changed that, because there are many mathematical functions with which we can represent the base color of a map. This allows us to save 5 tiles, which we can then use for other graphics. Above I already described how you can use a colored tile as a map. This will look like this. `cls` deletes all old data from the screen, `map` shows the map we draw in the map editor and `plr.spr` is the player sprite. what our character shows. Since I want to save myself the intermediate step of drawing each map over and over again, I want to display it *automatically*.

```
function _draw()
 cls()
 map()
 spr(plr.spr,plr.x,plr.y,
  plr.w,plr.h,plr.flp)
end
```

We change the line with `map()` to `rectfill(0,0,127,127,11)`. This will solve some problems. We don't have to draw the map as background. Since we want to have the map in one color, this is solved with one line of code. We can determine the size of the map ourselves and change it quickly. We save a total of 5 tiles because the color can be displayed in the source code. The colors are defined for each biome. The grassland `[11]`, tundra `[7]`, jungle `[3]`, swamp `[4]` and desert `[10]`. All in all, you should always look where you can save money in a small amount of space. If you ask yourself what you can do in 64Kb, you should definitely have a look at Pouet and be inspired by the projects there. You can also find many of the demos as videos on the internet. Search for *Demoscene*



In order to speed up the work we will first concentrate on the grassland. We will develop a prototype that we can then transfer to the other biomes. Now we ask ourselves what exactly should happen, because this helps us to concretize our game idea and automatically more questions will arise. Here is a list of the previous gameplay.

- The game starts
- The player is seeded into a randomly generated biom 1-5.
- The nature matching the biom is randomly set
- The player can move on the map

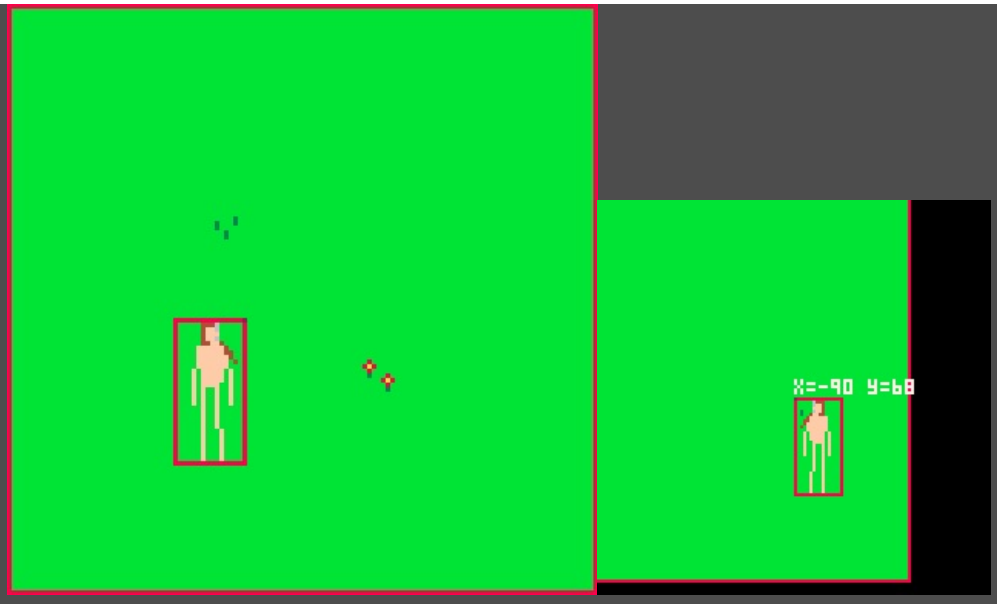
Let's think like real programmers working on a project. If we look at this list, we can see something. There are individual points associated with tasks. It almost looks like a recipe if you want to cook pasta with tomato sauce. Such a recipe is called the programming algorithm. An algorithm is an unambiguous rule of action for solving a problem. In our case, the problem is our goal, because the computer player is supposed to play our game. Since we are just learning how to program, we imagine this list as pseudocode. Even though teh Internet contains a lot of information about pseudocode and which rules to follow, there is only a concrete definition of pseudocode for me privately.

> *Pseudocode is an invented syntax that allows a programmer to solve an idea, problem or task by inventing an invented language to visually represent that thought. A pseudocode is considered to be concrete and meaningful if it can accomplish this task and its inventor can explain these other people exactly.*

```
Random Nature = x
Map = y

clear Screen()
create Random Map(x)
   if Map(x) then
      create Random Nature(y)
   else
      do nothing
   end
create player
end
```

At first this looks worse than it is and when we program in Pico8 we have a different look. It should be just an example how pseudocode could look like. In summary, the pseudocode is the list we have already seen above. This technique always helps me when I have a node in my brain and I can't get anywhere. This allows you to look at the problem or task from a different angle and get to the solution faster. But it also helps to ask further questions. In our project, for example, I ask myself the following questions. *How often is grass, flowers etc. displayed on the map? How big is the map? How can we prevent the player from "falling" out of the map?* These are all questions you might not have thought of before and now all of a sudden.



To solve the problem I read the Random Function article, exchanged myself with other Pico8 game developers on Twitter and watched a tutorial video about random numbers. So I had all the information I needed to solve this task. The final state so far is: The grass biom (not random yet) is created, then the elements from nature (grass, flower) are placed in the biom (only once at a random position) and in the last step the player is created. Of course, this is not yet what we want as a complete solution, but you have to divide the big task into small tasks to program a game. That's why I wrote at the beginning of the article that this is all work in progress and can change quickly. Let's get to the tasks for the future.

So far the biom is displayed statically. This means that the number is fixed. We also want to have this number as a random function so that we can switch between the 1-5 biomem. Also more than one flower and one grass bush should be displayed. We will have to take care of how many times something appears on the map. To display something randomly is a very powerful game mechanic, because it always shows the player something new and the game doesn't get so boring. Imagine a simple survival game in which you always spawn in the same place, with the same trees and the same stones. This will be boring in the long run. That's why I want to distinguish between 5 biomes. For this the suitable elements from nature should be set. Even if I am here again, it is illogical of the game world to let a snowball appear in a desert. Therefore, we make the elements of nature dependent on the biom that has been set.

As you can see at the top of the screenshot, I have a red frame that defines the size of our map. I set this to 127x127 to keep it clear. Everything has to be divided into small subtasks again and again. This red frame will later be the END OF THE WORLD. The player cannot cross this red line. Therefore we have to implement a collision detection. Since we have only a very small storage space available we have to limit the size of our game world strongly. But this is always the case in games and will not change so quickly, because the game world always depends on the physical storage space. So that we can move our character later in a larger world, we will install a camera that follows the player. This will give us a small working demo, but unfortunately we still have to manage without a game goal.

As you can see in the gif animation, everything looks a little different than I just described it. There's a reason for that. I watched the tutorial video Pico-8 Adventure Game Part 2 - Sprites, Maps and Cameras by Rik Gross and was convinced, that it's better to install a camera in the game first. Since we have the focus on the character of the player, he is always to be seen in the middle. This is the case in a lot of computer games, but sometimes you see an exception. In addition, I've also had the coordinates of the exact position of the player displayed. I'm thinking about the idea of including this in the actual design to simulate the latitude and longitude. In a Survival game I can certainly imagine this to be interesting.